REMARKS

Claims 1-12, 14-30, and 32-38 are pending in the present application. Claims 1-3, 9, 11-12, 17, 19-21, 27, 29-30, 35 and 37-40 have been amended and no new matter has been added. Claims 8, 26, and 37 have been cancelled. Claims 13 and 31 have previously been cancelled. New claims 39-60 have been added. Accordingly, claims 1-7, 9-12, 14-25, 27, 30, 32-36 and 38-60 are now pending in the present application. Applicants find support for the claims generally throughout the specification, specifically on pages 5-14 and in Figures 1-5. The claims as presented herewith have been approved by Examiner Todd D. Ingberg, prior to the filing of this amendment under 37 C.F.R. § 1.312. No new matter has been added.

This preliminary amendment is responsive to the Notice of Non-Complaint Amendment mailed July 26, 2007.

This amendment being filed herewith supersedes all prior amendments and is consistent with the draft amendment as sent to the Examiner on April 9, 2007, and subsequently approved in his Summary of Interview on April 16, 2007.

Present Invention

An improved process for providing an image of software installed on a computer system is disclosed. The process includes the steps of deconstructing an existing image and creating one or more modules from all or part of the image utilizing either install information or uninstall information. To deconstruct the image, the image is scanned to identify at least one portion of the image to be modularized. At least one portion of the image is then extracted, and at least one module is generated from that portion of the image. The module can then be formatted for use in a new image or part of a new image to be used with a particular software program, such as with a

hardware-independent imaging tool or with other hardware-independent application software. An advantage of making an image modular is that it allows hardware-independent software programs (e.g., operating system, commonly used application software) to be abstracted or separated from hardware-dependent software programs (e.g., device drivers, hardware-dependent software). Modules can be added or removed from an image as needed, or can be combined to create new modular-based images.

Claim Rejections – 35 USC § 102

The Examiner states,

> Claims 1-38 are rejected under 35 U.S.C. 102(b) as being anticipated by DERIVE: A Tool That Automatically Reverse-Engineers Instruction Encodings, Dawson R. Engler et al., ACM, 2000, pages 12-22.
>
> DERIVE anticipates a method for providing an image of software installed on a computer system, the method comprising the steps of:
>
> (a) deconstructing the image into at least one portion (Derive, Abstract, page 1, Reverse Engineering – installed software); and
>
> (b) creating at least one module from the at least one portion of the image (Derive, Conclusion, page 19, instruction encoding and page 22, encoding structure, Figure 5 – emitter specification).
>
> (c) formatting at least one module for use in a new image or at least a portion of a new image.
>
> Examiner note: When taking the reference as a whole, please, look on page 14 Figure 1 at the information flow for a detailed view. DERIVE solver produces encoding description and the emitter generator feeds the Instruction emitter, the presence of JIT is the Just In time Compiler which produces the new image in cooperation with the Instruction emitter. Also, please look at the bottom of page 18 one of the last sentences on Linkers "...for only a few specific type of machine-dependent information, derived by feeding appropriate inputs to existing assemblies and linkers." The Linker by definition formats input into images. That is the role of the linker.

Applicants respectfully submit that the independent claims 1, 17 and 37-39 are not anticipated by the DERIVE reference. For ease of review, claim 1 is reproduced below:

> 1. (Currently amended) A method for providing an image of software installed on a computer system, the method comprising:

> deconstructing the image into at least one portion, wherein the at least one portion of the image comprises an operating system and code, wherein the code is selected from a group consisting of a set of drivers, a set of utilities and application software;
>
> creating at least one module from the at least one portion of the image utilizing information, wherein the information is selected from a group consisting of install information and uninstall information; and
>
> formatting the at least one module for use in a new image or at least a portion of a new image.

Applicants submit that DERIVE discloses a method of reverse-engineering instruction encodings from pre-existing software (the system assembler) and uses the information extracted to construct dynamic linking libraries, object-level sandboxers, executable optimizers, and linkers. Accordingly, DERIVE discloses reverse engineering instructions from software. In contrast, the present invention comprises deconstructing an image into at least one portion. As described in the specification, page 6, lines 11-13, an image comprises an operating system, at least one of a set of drivers, a set of utilities and/or application software. A portion of an image that is deconstructed from an image is clearly different than the reverse engineering of instructions as disclosed in DERIVE. In fact, DERIVE may be used as a portion of the recited invention by reverse engineering specific instructions but it is not capable of deconstructing an image into at least one portion.

In addition, a system in accordance with DERIVE reference copies a program from one instruction set to another instruction set. In contrast, in the present invention a module is created utilizing either install information or uninstall information from the at least one portion. Therefore, the creating of a module includes performing tasks required to install a new image on another computer system. For example, as stated in the specification at page 9, line 16-page 10, line 16,

> "There can be one or more portions and one or more modules generated from each portion depending on the specific application. In the preferred embodiment, the module is generated using uninstall code, also referred to as uninstall "scripts," which are commonly used to remove an installed software program. To generate the module from the uninstall scripts, the uninstall scripts

are first scanned/searched and analyzed in reversed order to determine the actions that have taken place to install the software. The uninstall scripts are typically stored in an uninstall file, in a registry, or in the OS software and accessed from a dynamic-link library (DLL). The uninstall scripts typically include data such as application specific actions, decrement reference counts, shared DLL files, removed registry keys, pointers, links, files copied, and/or moved, etc.

The module can then be installed onto a computer system or processed by an imaging tool by using install scripts that correspond to the uninstall scripts. The install scripts can be determined from information from the uninstall scripts in combination with log information related to the OS during an original installation. When a software program is installed under an OS, the OS maintains a log of actions taken during the installation process. For example, the log includes information on changes to the OS software. Such changes can include, for example, newly shared DLLs reference counts, removed tags, decremented reference counts, etc. Such information can be used to configure the generated install scripts. The install scripts are ascertainable because the install and uninstall procedures are standardized. Accordingly, existing information in the image can be used in a reverse engineering process to create install scripts from the uninstall scripts. The install and uninstall scripts can be stored in a location specified by the user or in a default location such as with the files needed by related software programs."

As seen from the above, the invention as recited in the independent claims as well as the claims dependent thereon are not taught or suggested by the DERIVE reference because the DERIVE reference is directed to reverse engineering or copying of instructions to allow the instructions to be transferred from one instruction set to another instruction. As stated above, the recited invention provides at least a portion of an image which is clearly different from an instruction set. Furthermore, in the recited invention at least one module is created utilizing either uninstall information or install information from at least one portion of an image. DERIVE neither teaches nor suggests an equivalent process. Accordingly, this cooperation of elements are not taught, suggested or contemplated by the DERIVE reference.

Accordingly, claims 2-12, 14-18, 20-30, 32-36 and claims 40-54 are allowable since they depend from allowable base claims as well as for the above-stated reasons.

Claim Rejections – 35 USC § 103(a)

The Examiner states,

> Claims 5, 7, 14-16, 18, 23, 25, 32-34 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over DERIVE in view of Modular Type-Based Reverse Engineering of Parameterized Types in Java Code, Dominic Duggan, ACM, 1999, pages 97-113.
>
> Since, it is not clear if the independent the Applicant is claiming is from the input of the output of reverse engineering the Examiner has elected to reject the following claims under 35 U.S.C. 103(a).
>
> Motivation to Combine DERIVE and JAVA
>
> DERIVE teaches the emitting of C code (DERIVE, page 22). C code is not universally known to be platform independent. It is JAVA who teaches a well known platform independent language. Therefore, it would have been obvious to one of ordinary skill in the art to combine DERIVE and JAVA, because reverse engineering for a language like JAVA which is platform independent by the implementation of a virtual machine, would make a reverse engineering tool more flexible.

Applicant submits that the arguments hereinabove with respect to the DERIVE reference apply with equal force to this rejection since these claims depend from allowable base claims.

The JAVA reference describes a language independent platform but the combination of JAVA reference and the DERIVE reference provides for the reverse engineering of instructions utilizing a language independent platform. For the above-stated reasons, this is clearly different from the invention as recited in the above-identified claims.

Accordingly, claims 5, 7, 14-16, 18, 23, 25, 32-34 and 36 are allowable over the cited references either singly or in combination for the above-cited reasons in the above-identified claims.

New Claims

Applicants have added a new independent system claim 39 that has similar limitations to that in method and computer readable medium claims 1 and 19.

Accordingly, claim 39 is also allowable over the cited reference for the above-mentioned

reasons. Furthermore, claims 40-54 are also allowable since they depend from an allowable base claim as well as for the above-stated reasons.

Independent computer readable medium claim 38 has been amended and new independent system 55 has been added to further define the scope and novelty of the present invention. Specifically both claims specifically recites "creating the at least one module from the at least one portion of the image utilizing uninstall code". Applicants respectfully submit therefore that these claims are allowable for the same reasons as stated for independent claims 1 and 19.

New dependent claims 56-60 are added to further define the scope and novelty of the present invention.

On the basis of the above remarks, Applicant respectfully requests that the above amendments be entered. If the Examiner has any issues with entering the above amendments, the Examiner is respectfully requested to contact the undersigned at the number listed below.

Conclusion

For the above-identified reasons, Applicant respectfully requests reconsideration and allowance of claims 1-7, 9-12, 14-25, 27-30, 32-36, and 38-60 as now presented.

Applicant's attorney believes that this application is in condition for allowance. Should any unresolved issues remain, Examiner is invited to call Applicant's attorney at the telephone number indicated below.

Respectfully submitted,

SAWYER LAW GROUP LLP

August 1, 2007

/Joseph A. Sawyer, Jr./

Joseph A. Sawyer, Jr.
Attorney for Applicants
Reg. No. 30,801
(650) 475-1435